September 31, 1979 Vol. I No. 10

KEYBOARD ETC., STUFF: Latest words on the Bally Programmable Keyboard will be on p. 84.

KEYPAD SUBSTITUTE originally reported on p.47, has been completed by Ed. Larkin, who offers details as to how he did it in his ad this issue. The idea provides the user with a full size keyboard to do the same functions as the keypad, only in a more convenient form for most of us. You still have to punch two keys to get a letter,etc, as there is no built-in delay circuit that would add to the cost.

KEYBOARD/MEMORY UNIT mentioned on p.69 is getting a little closer. I am in the process of receiving a price quotation on the following: A memory board with 32K of RAM of which 16K is dedicated to the operating system which will be cassette - loaded at 1200 baud (about two minutes). Serial and parallel ports and expansion provisions to be included. Compatibility with the Jameco JE610 keyboard is expected. The operating system will be very sophisticated and unique. Details on the above are being included in this mailing to those who have responded to my survey. This hardware will not be generally advertised.

SERIAL NUMBERS are being collected against the day that a use is found for them. There are three Model Numbers, depending on the sales outlet:

BPA 1000 is sold by Montgomery Ward catalog
BPA 1100 is sold through retail/TV stores
BPA 1200 was sold by JS&A

The only real difference between them is the logo on the plastic cover. A small difference came about with the white case units that were marketed with only two hand controllers.

GAME MODIFICATIONS An addition to SLOT MACHINE by Phil Shafer takes care of the case where you win but are still short

```
1511  IF M<=Ø  M=M+N; GOTO 1515
```

Mike Fink says the following addition to CHECKERS will allow you to see the move immediately

```
1615  IF T>Ø  GOSUB 2ØØØ
```

Wayne Dunning notes that BLACK BOX should have a comma in line 145 after the first B and in front of the semicolon. Bob Strand indicates that line 490 should have a M=-1 instead of M=-M.

SIMON corrections of p. 45 have an inadvertent error of mine in that I added three GOTO 170 statements and then deleted 170 ! It should be retained `170 FC=Ø; NEXT X`

REVIEWS OF GAMES etc., was mentioned on p. 76. I have received a number of names of potential reviewers so if any of you workers would like to have your outputs reviewed objectively on the basis of such categories as - level of challenge, originality, educational value,etc., plus some subjective comments, send your material to one or another of the below-listed gentlemen, and make your own arrangements. Include all documentation,etc., that would be sent to a purchaser. I in turn will print the reviews editing only for space limitations. We are working on a form grading system and will run a copy in the ARCADIAN for all to see.

VOLUNTEERS:				
	Steve Wilson	18015 Sally Ave.	Cleveland OH	44135
	Don Daniels	3 Apex Rd	Melville NY	11746
	Bill Rueger	336 Beach 38th St	Far Rockaway NY	11691
	Phil Shafer	3708 Big Horn Trail	Plano TX	75075
	Dick Hauser	635 Los Alamos Ave	Livermore CA	94550

TUTORIAL - USING THE BALLY BASIC TEXT AREA

David Ibach
19553 Dartmouth Pl.
Northville, Mich. 48167
Sept. 3, 1979

Now that we know where the text of our program is stored (A000 thru A707 or decimally -24576 thru -22777), there are several uses we can make of this information:

1. Storing data in the text
2. Writing self-modifying code
3. Storing machine code in the text

These uses require PEEKing and/or POKEing with the %(addr) construct as described on page 19. (Jan. 13, 1979 "Arcadian").

Storing data in the text may be something you want to consider as a substitute for the DATA statement available in more powerful BASICs. Since the data is in the text area, it will be written on the tape when you store your program. Simply branch around the data in your program so BASIC doesn't try to execute it.

Here's an example you might find useful. Suppose you want to play a tune in your program and you don't want to PRINT characters to do it. Loading MU serially with the desired tones (a series of MU=dd statements) is costly in terms of memory used. The string variable may not be available, or even it it is available, it is not easy to store its values on tape. Why not write your tune as part of the program text. That way you will be storing it on tape with your program. And since Bally Basic stores one character per byte, you get optimum use of memory. Try this:

```
 1 GOTO 5
 2 "3012312340234234506034123532 1
 5 NT=15
10 FOR Y=-24568 TO -22777 STEP 2
20 Z=%(Y)÷256;IF RM=13 GOTO 50
30 MU=RM;IF Z=13 GOTO 50
40 MU=Z;NEXT Y
50 NT=3;STOP
```


For this to make sense, let me explain how Bally Basic stores its text. To begin, statement numbers occupy 2 bytes of memory regardless of the number of digits in the statement number. However when a statement number is referenced (as in GOTO 5 of the first line), the number of memory bytes used is equivalent to the number of digits in the statement number. In fact, all keystrokes in a statement (minus the bottom row of qualifiers on your keypad) require one byte of memory each. Thus keywords such as GOTO and INPUT use only one byte each. In addition, the GO keystroke at the end of every statement uses a byte of memory. It is stored as a 13 (hexadecimal 0D) and explains how the end of the song is detected in line 20 or 30.

Here then is how the beginning of this program is stored in the text area:

Location	Content	Comments
-24576	Stmt. No. 1	Occupies 2 bytes
-24574	GOTO	Occupies 1 byte
-24573	5	Hex 35 or decimal 53 represents character '5'
-24572	GO character	Hex 0D or decimal 13
-24571	Stmt. No. 2	Occupies 2 bytes
-24569	" (quote char.)	Inserted so Basic could distinguish the tune from stmt. no. 2
-24568	3 (beginning of tune)	This location is initial Y value in stmt. 10

Self modifying programs are fun to play with. There are dangers involved since the logic is more complex. And to restart a program you'll probably have to reload it in its original form. Nevertheless, the technique does have its applications.

As an example of self modifying code, key in the following program. After execution notice how line 10 has changed.

```
10 GOTO 20
20 PRINT " FIRST LIST ,1:"
30 LIST ,1
40 %(-24573)=12342
50 GOTO 10
```

more . . . .

arcadian

```
60 PRINT " SECOND LIST ,1:"
70 LIST ,1
80 STOP
```

I've tried putting machine code in the text but so far I've been unable to execute it there. Bally Basic will honor a call to machine code in other memory locations (eg. the line input buffer), but the keyboard locks up when the call address is within the text area. If anyone can shed light on this I'd like to hear.

In closing, just a few notes on these techniques to help you avoid trouble:

- Remember each PEEK or POKE references 2 bytes of memory (hence STEP 2 in line 10 of first program above)

- Since memory addresses are expressed as negative numbers (starting with -24576) you advance by decrementing the absolute value

- Page 12 of the Bally Basic Hackers Guide tells you in decimal how each character is represented internally, including the keywords

- If the values you store in the text area are not recognizable as characters to Basic, they will load with question marks, but the load should be accurate.

- If you have a program in memory and want to know the address of, say, statement 5200, enter the following commands directly:

```
>FOR N=-24576 TO -22777;IF %(N)#5200NEXT N
>PRINT N
```

- Remember the GO character at the end of every line when counting bytes.

```
 1 .SPACE WAR
 2 .  D IBACH  7-79
10 GOSUB 100
20 IF TR(1)=1GOSUB 200
30 IF H#0GOSUB 800
40 IF JY(1)#0G=JY(1);GOSUB 300
50 IF TR(2)=1GOSUB 400
60 IF H # 0 GOSUB 800
70 IF JY(2)#0G=JY(2);GOSUB 500
80 GOTO 20
100 CLEAR ;BC=0;FC=0;N=RND (12)
    +9;C=0;D=0;GOSUB 700;FOR M=
    1TO N;X=RND (150)-75;Y=RND
    (80)-40;BOX X,Y,1,1,1;NEXT
    M;A=RND (76)-38
110 B=RND (76)-38;H=0;FC=7;
    RETURN
200 H=0;G=KN(1)÷(-3);LINE -60,
    A,4;MU=7;LINE 60,G,3;LINE
    -60,A,4;LINE 60,G,3;IF G<B
    +5IF G>B-5H=1
210 RETURN
300 A=A+5*G;IF A>40A=A-5
310 IF A<-40A=A+5
320 RETURN
400 H=0;G=KN(2)÷3;LINE 60,B,4;
    MU=2;LINE -60,G,3;LINE 60,
    B,4;LINE -60,G,3;IF G<A+5
    IF G>A-5H=2
410 RETURN
500 B=B+5*G;IF B>40B=B-5
510 IF B<-40B=B+5
520 RETURN
700 CX=-65;CY=40;PRINT C,D
710 IF (C=15)+(D=15)=0RETURN
720 NT=0;CX=-8;CY=5;PRINT
    "GAME";CX=-8;CY=-5;PRINT
    "OVER";NT=3;FC=0;BC=7;STOP
800 BC=7;&(23)=255;&(21)=255;
    BC=0;FC=0;IF H=1GOTO 810
805 BC=7;FC=7;S=-60;T=A;GOTO
    815
810 BC=7;FC=7;S=60;T=B
815 BC=0;FC=0;IF T>33T=33
817 BC=7;FC=7;IF T<-34T=-34
818 BC=0;FC=0;LINE S-5,T+10,4;
    LINE S+5,T-10,1
820 BC=7;FC=7;LINE S+10,T+5,4;
    LINE S-10,T-5,1
822 LINE S+5,T+10,4;BC=0;FC=0;
    LINE S-5,T-10,1
824 LINE S+10,T-5,4;LINE S-10,
    T+5,1;FC=7
830 FOR N=1TO 24;BOX S,T,N,N,2;
    NEXT N;&(23)=0;&(21)=0
840 CX=-65;CY=40;NT=0;PRINT
    "          ZZZZZAP!";NT=3;
    IF H=1C=C+1;B=RND (76)-38;
    GOTO 860
850 D=D+1;A=RND (76)-38
860 GOSUB 700;H=0;RETURN
```

This is a two player game. Player one owns an invisible space ship on the left hand side of the screen. He can move it up or down by pushing his joystick forward or back. When he pulls his trigger, he shoots across the screen at the invisible ship of player 2. A shot also exposes his position on the screen. The invisible ship of player 2 is on the right side of the screen and is similarly controlled. Theknob setting determines the angle of the shot.

SPEEDUP TO TAPE A note from Ed Mulholland says that increasing the machine's speed by decreasing the Note Time will work for tape transcriptions. Ed reduces NT to 1 in the directions to transcribe- :PRINT;NT=1;LIST saying that if NT is ϕ, there won't be any audio. But Ed Larkin has reported that if the NT is put ahead of the other commands, it will work for him - NT=ϕ:PRINT;LIST. See what works for your machine.

DIVISION with results in non-decimal format was run by Marc Gladstein for those who would like to see the quotient printed with the remainder continued as a fraction. The gist of it is -

```
10  INPUT "D1=" X
20  INPUT "D2=" Y
30  Q= X÷Y; R=RM
40  PRINT "QUOTIENT = "
50  PRINT #1,Q,;IF R PRINT #1," ",R,"/",Y
```

SUBSCRIPTION RENEWAL TIME is coming up. Because of the timeless(?) value of most of the material of the ARCADIAN, and because I don't have any bookkeeping capability (it would be nice to have a computer), all subscriptions are on a volume basis, one year from November to October, and everyone receives all the back issues in a lump at the time he/she subscribes. I am now soliciting subscriptions for Volume 2, to start in November of 1979, at the rate of $10. The issues will again be guaranteed as bimonthly, with added issues as material becomes available, the same as was true for 1979. I expect that with the keyboard/memory that we are working on now will generate a lot of acitivity in its own right as will peripherals. Tiny BASIC will continue to surprise us, and we are developing some hardware modifications to the basic machine to make it better, so there seems to be a lot of material that will come forth.

TUTORIAL on text area by Dave Ibach includes a game that sounds interesting.I have not had the opportunity to try it out as yet. In the second line of Dave's tutorial is the indication of storage being located at -24576 thru -22777. This serves as a correction to the table I printed on p. 34, "Text Area".

DICTIONARY by Steve Walpole on p.82+ provides you with a conversion between some commands used in other BASIC language programs and the TinyBASIC of Bally. From a format standpoint, Steve first gives the general command and a short statement about it, and then how to do the same thing in TinyBASIC, or as close to it as possible.

SAMPLE PAGE shown at the top of p.83 is probably understandable only to those who can read assembly language. It is my intent to have the most interesting of these pages "transcribed" into English for the rest of us, and also to have some programs developed utilizing these for all of us.

SUGGESTIONS,etc. I have a number of programs on hand for the next issue. My problem is the transcription of them from whatever form they are in into one that is legible,especially after reduction (usually to 75 or 50%). I would appreciate program listings to be either: typed, or clearly hand printed on a form such as that provided by Chuck Thomka. Most company forms have lots of little bitty boxes that each letter/character fits into and/or colored sections that do not make for good clear reproduction. Please include explanations. Anything that can be directly printed in the ARCADIAN should be typed unless your handwriting is Spencerian or you use the Palmer Method. If I receive listings which have to be transcribed, they will be sent back to the originator for proofreading after transcription/reduction. I assume that those that arrive all ready for printing will have been proofed.

The following are my comments on the PX function:

The possibility of the PX function as a location sensor seems reasonable if you only have to monitor whether a player (meaning a visible marker such as a BOX) is at a given location or not.

I have enclosed a simple program which uses the PX(X,Y) function as a location sensor in the manner of a trap being sprung. Ten traps (explosive mines, invisible enemy ships, etc.) are set randomly, and if the player moves over any of the trap locations, he is trapped (caught, exploded, etc.).

I don't see how this function could be used in two-player games in general, since only two conditions can exist: PX(X,Y)=0 or PX(X,Y)=1. In many games, monitoring is needed for three functions: PLAYER #1 (black), PLAYER #2 (white), and neither player. This is the case with most board games.

Two-player games where both players have black markers could use PX to monitor both players, since only one player can move at any one moment.

Also, PX could be used to monitor the intersection of two player markers if they were reverse BOX markers. There intersection would then be white if the markers are black, and the PX function would equal 0 when they intersected.

Sincerely yours,

Steve Walters

Steven L. Walters

556 Langfield
Northville, Mi. 48167

PROGRAM USING PX(X,Y) AS A LOCATION SENSOR

```
10  CLEAR                                  COMMENTS
20  FOR N=1 TO 19 STEP 2                   Sets 10 location sensors at
30  @(N)= RND(100) - 50                    PX (@(N), @(N+1))
40  @(N+1)=RND(60) - 30
50  NEXT N
60  (see optional section)
90  X=-70; Y=0                             Start location for box marker
100 X=JX(1)x3+X
110 Y=JY(1)x3+Y
120 IF X< -70 X=-70                        Sets movement limits on box marker
130 IF X>70  X=70
140 IF Y< -35  Y=-35
150 IF Y>35  Y=35
160 BOX X,Y,5,5,1                          displays box (player marker)
200 FOR N=1 TO 19  STEP 2                  Test if marker is over any
210 IF PX( @(N),@(N+1) )=1  GOTO 300       PX sensor location
220 NEXT N
230 BOX X,Y,5,5,2                          Erase marker, repeat
240 GOTO 100
300 BOX X,Y,7,7,3                          Visual feed back for sensor response
310 BOX X,Y,9,9,3                          (trap appears surrounding
320 PRINT "CAUGHT!                          marker)
330 STOP
```

Option: to display sensor locations visually as marker is moved about:

Add the following:

```
60   GOSUB 400

400  FOR N=1 TO 19 STEP 2
410  BOX @ (N), @ (N+1), 11, 11, 1
420  BOX @ (N), @ (N+1), 13, 13, 3
430  NEXT N
440  RETURN
```

DICTIONARY of TRANSLATIONS from BASIC to Bally BASIC

AND - The AND statement allows for more than one condition to be placed in a single IF statement.

```
10 IF A=0 AND B=0 GOTO 120
```

The program will branch to 120 only if A=0 AND B=0. With Bally BASIC use a second IF statement in the place of AND or put the conditions in parentheses.

```
10 IF A=0 IF B=0 GOTO 120
   OR
10 IF (A=0)+(B=0)=2 GOTO 120
```

(See page 52)

ASC-CHR$ - The ASC function converts any given character into its ASCII code number while the CHR$ function does just the opposite, converting an ASCII code number into its equivalent character.

```
10 A=ASC(A)
20 PRINT A
30 A$=CHR$(65)
40 PRINT A$
RUN
65
A
```

With Bally BASIC, the advantage of turning a letter into a number is because you can't store a letter in a string or counter only a number. Then by using the TV function you can call upon a number to be changed into a letter and displayed on the screen.
Use the KP function instead of ASC:

```
10 K=KP
20 PRINT K
30 TV=K
RUN

   65
A
```

In line 10 of the example, the computer waits until a character is typed in on the keypad. Then it automatically converts that character into its ASCII code number and stores it in the K counter. In line 20 the computer prints the value of K and in line 30 the computer converts the value of K into its ASCII character and displays it on the screen. If you do not want to have to input the same letter every time you run the program look up the ASCII code number you want on page 16 in the Decimal column and store that number directly inthe counter or string.

```
10 K=65
20 PRINT K
30 TV=K
RUN
   65
A
```

INT - This function removes the decimal from a number and returns only the whole number.

```
10 A=4÷3
20 PRINT A
30 PRINT INT(A)
RUN
1.3333333
1
```

Bally BASIC does this automatically so INT or anything else is not necessary.

```
10 A=4÷3
20 PRINT A
RUN
    1
```

LET - LET assigns a variable or string to any given value.

```
10 LET A=45
```

LET is not necessary with Bally BASIC. Just omit the statement LET.

```
10 A=45
```

NOT - NOT is used with the IF statement.

```
10 IF NOT A GOTO 120
```

If A=0 the program will branch to 120. If A equals any other positive or negative number and the program will resume with the next line number. With Bally BASIC use:

```
10 IF A=0 GOTO 120
```

ON-GOSUB - This statement is used for multiple branching.

```
10 ON A GOSUB 120,200,340,500
```

In the example, the program will GOSUB 120 if A=1; 200 if A=2; 340 if A=3; and 500 if A=4. There are a couple of ways this can be done with Bally BASIC. The first one is where you have to use many lines.

```
10 IF A=1 GOSUB 120
20 IF A=2 GOSUB 200
30 IF A=3 GOSUB 340
40 IF A=4 GOSUB 500
```

This takes up too many bytes to be practical so there is a better way. Space the line numbers of the subroutines evenly apart (200 in the example) so that the product of Ax200 will guide the program to the correct line. Try the sample program below:

```
 5 CLEAR
10 A=0
20 IF JY(1)=1 A=1
30 IF JY(1)=-1 A=2
40 IF JX(1)=-1 A=3
50 IF JX(1)=1 A=4
60 CY=0:CX=0
70 IF A=0 PRINT "*":GOTO 10
80 GOSUB Ax200
90 GOTO 10
200 PRINT "↑";RETURN
400 PRINT "↓";RETURN
600 PRINT "←";RETURN
800 PRINT "→";RETURN
```

Move the joystick to control the arrow.

ON-GOTO - Works the same way as ON-GOSUB except using the GOTO statement.

OR - Works the same way as AND. allowing more than one condition to be placed in a single IF statement.

```
10 IF A=0 OR B=0 GOTO 120
```

Except with the OR statement the program will branch to 120 if A=0 OR B=0 as with AND the program would branch only if A=0 AND B=0. With Bally BASIC use:

```
10 IF (A=0)+(B=0) GOTO 120
```

(See page 52)

READ-DATA - This statement is used when large amounts of variables and/or strings are to be assigned values.

```
10 READ A,B,C,D
20 DATA 25.40.44.60
```

When the program reaches a READ statement, the computer searches for the first DATA statement. takes the first value of that statement and assigns that value to the first variable of the READ statement. If there are any more variables inthat READ statement the computer will then search for the second value of the DATA statement and assign that value to the second READ variable. etc. Therefore, in the example. A=25; B=40:C=44:and D=60. To do this with Bally BASIC. each variable will have to be assigned individually but they can be placed on the same line.

```
10 A=25:B=40:C=44:D=60
```

REM - REM stands for "remark" and it means just that. It has no special function except to provide an in-program documentation of the program.

```
10 REM THIS PROGRAM SIMULATES
20 REM NEGATIVE GRAVITY IN SPACE
```

With Bally BASIC just use a period (.) in the place of REM.

```
10 .THIS PROGRAM SIMULATES
20 .NEGATIVE GRAVITY IN SPACE
```

Since the Bally system does not have alot of memory, it is best to leave out these lines unless your program is short enough to allow it.

TAB - TAB refers to how many spaces from the left side of the screen to print before printing the word(s) following it.

```
10 PRINT TAB(5) "COMPUTER"
```

With Bally BASIC enclose the number of spaces in the quotes along with the word to be printed or use the CX function.

```
10 PRINT "     COMPUTER"
20 CX=-47:PRINT "COMPUTER"
RUN
     COMPUTER
     COMPUTER
```

To determine the value of CX, start with -71 for 1 space and add 6 for each additional space. So for 2 spaces CX=-65, 3 spaces CX=-59, etc.

SYMBOL TRANSLATIONS

Multiplication sign- * to x
Division sign - / to ÷
String symbol - $ to @
Colon (:)-This symbol is used in most BASICs to allow more than one command per line. With Bally BASIC the semi-colon (;) is used. And in other versions it can be a slash (/) or back-slash (\). Be careful not to mistake these for a division sign.

THEN - This means the same as GOTO. It is usually found in an IF statement. Just replace THEN with GOTO.

THIS IS A PAGE TAKEN FROM A DOCUMENT THAT I HAVE WHICH INCLUDES ABOUT 140 PAGES OF ROUTINE DESCRIPTIONS SIMILAR TO THE SAMPLE, PLUS ABOUT 200 PAGES OF OBJECT CODES FOR APPARENTLY ALL THE ROUTINES IN THE MACHINE. IF YOU ARE INTERESTED IN THE CONTENTS OF THIS VOLUME, DROP ME A LINE OR CALL ME AND WE'LL DISCUSS IT.

SCREEN ALPHANUMERIC DISTIM

DISPLAY TIME

Calling Sequence:
```
SYSTEM DISTIM
   or
SYSSUK DISTIM
DEFB   (X co-ordinate)
DEFB   (Y co-ordinate)
DEFB   (options)
```

Arguments:
DE=X,Y co-ordinates
X =Options (see note below)
IX=Alternate Font Descriptor (not loaded)

Outputs:
DE=Updated

Description:
This routine displays the system time (GTMINS,GTSECS) at the co-ordinates specified in the form MM:SS, where M=minutes, S=seconds. Seconds are optional.

Notes:
The small character set is used and one level of enlarge factor is permitted.
Options are the same as the alphanumeric display routine except that bit 7=1 to display colon and seconds; bit 7=0 to suppress colon and seconds.

ADS

2-player BATTLESHIP; 1 player JOTTO/SENSOR (two 120-word versions available-general words,and expert); variable size/difficulty MASTERMIND. All for $6 your tape or $7 his tape. Don Daniels, 3 Apex Rd. Melville NY 11746

Bally BASIC $30; Interface $30;Brickyard/Clowns,Blackjack/Poker, SpeedMath, SeaWolf/Missile @$15 ea. 8 Handcontrollers @$5 ea. J.Jones, 723 S. Gardena, Rialto CA 92376

LISTINGS only for COMPUTER CRAPS $2; SLOT MACHINE $2; RUSSIAN ROULETTE $1; SPELL'N'SCORE $1.50; CHECKBOOK BALANCER $1.50 or $7 for all. Also Service on hand controllers. S. Walpole, 11480 Beirut Ct. #204, Sappington MO, 63126

KEYBOARD in parallel with existing keypad: plans,specifications and photos $10.ppd. Ed Larkin, Outlet Rd. Hallowell, ME 04347

HARDWARE ITEM!-JOYSTICK CONTROLLER, a true joystick (2-100K pots), 360 deg. rotation, with two RS-232 connectors, black plastic case, and 10 MICROSWITCHES!! This is a multi-controller device, comes with software on tape W/listing & instructions on writing your own programs for it. $34.95 (+$3 p&h) available Oct 22. Write for details. Also, XY TUTORIAL package, for exclusive controlling of graphics, 12 pages +software on tape with SIX programs, listings included. $9.95. NEW ITEMS-SEBREE'S COMPUTING, TIM HAYS, 456 Granite,Monrovia CA,91016

DEALER selling out all stock on Bally-games,Basic,etc., all items at our original cost. Video Environment +, Inc. 580 New Loudon Rd. Latham NY 12110

BALLY ADD-ON I've kept this space open hoping for a last-minute official word, but I did not get any and time is short. What I've heard from various unofficial sources is that the FCC did allow the TI request which provides relief in the TVI areas(the news release has yet to come out). Whether Bally will react to this in a positive manner is a question. My sources are all down and think that chances are very slim that any Level III hardware will actually be produced. Many dealers have given up the line, as have some distributors. I hope to have some definitive news in the next issue, which by the way will be the last of Volume I.

ARCADIAN
Robert Fabris, typist
3626 Morrie Dr.
San José, CA 95127

FIRST CLASS